A VLSI Artwork Legalization Technique Based on a New Criterion of Minimum Layout Perturbation

Fook-Luen Heng†, Zhan Chen‡, Gustavo E. Tellez†

† IBM Corporation
32-138, T.J. Watson Research Center,
P.O.Box 218
Yorktown Heights, NY 10598
Email: heng, gus @ watson.ibm.com

‡ Dept. of ECE
Univ. of Massachusetts
Amherst, MA 01003
Email: zchen@biz.ecs.umass.edu

Abstract

In this paper we propose a novel VLSI artwork modification technique based on the concept of a *minimum layout perturbation*. Layouts are designed so that minimum design rules must be satisfied. Often layout processes such as custom layout methodologies and design rule migration activities introduce design rule violations in layouts. A minimum layout perturbation defines a minimum cost change to a layout, such that the resulting layout satisfies all design rules. We formulate the minimum perturbation cost with the objective of preserving as much as possible the geometric and topological features of the original layout. The proposed *minimum perturbation problem* formulation is transformed into a linear programming problem with special structure. We exploit the structure of the problem to propose efficient algorithms that solve the problem. We also propose and implement a practical graph-based simplex algorithm, which we compare to a commercially available linear programming package, resulting in more than 40X performance improvements in some cases. Finally, the proposed methods have been implemented and used in real life problems, for example in the technology migration of data path macros and a 300-cell gate array library.

1. Introduction

The generation of ground-rule correct VLSI layouts, also known as artwork, has been subject to a large body of research. Automatic techniques exist for automatic generation of layouts, such as placement, routing, and compaction. However many layouts, so-called *custom layouts*, such as memory cores and microprocessor data path macros are still designed manually. For these layouts very few effective automation techniques exist, thus making their creation a manual, error-prone process. In addition, layout methodologies such as design rule migration techniques routinely introduce ground-rule violations as part of a design shrink step. The problem that arises from these layouts is to remove these design rule violations with minimal changes to the layouts.

In this paper we investigate the problem of minimally modifying a starting layout with design rule violations, so that the resulting layout is ground-rule correct. As far as we know, the problem of correcting design rule violations with minimal perturbation of a layout has not been posed or formalized. Previous work on these types of problems has focused on layout compaction techniques. For a survey on these techniques see [1]. Unfortunately layout compactors consider trade-offs between only a few design objectives: mainly area, wire length and yield. Real layouts, however, must often deal simultaneously with complicated objectives like performance, power, reliability, yield, porosity and constraints imposed by design style. In compaction these other objectives must be captured in the form of layout constraints, which is a non-trivial, time-consuming and often frustrating task. Another problem facing the traditional compactor is when the input set of constraints cannot be satisfied simultaneously. The situation can be detected by finding a cycle of positive length in the input constraints set. In the presence of a positive cycle, there is no solution to the longest path problem (no feasible solution) and thus no obvious partial solution is provided. There are methods to provide partial solution in the presence of positive cycles for compaction in the traditional sense [4]. We give a practical partial solution in the context of correcting rule violations.

The outline of this paper is as follows. In Section 2 we motivate the work through a real life example. In Section 3 we propose a graph-based formulation of the problem with a convex objective function. In Section 4 we propose a transformation of the objective function that yields a linear objective, and then propose several algorithms that solve the linearized objective. In Section 5 we show some comparisons between the different proposed solutions, and in Section 6 we present some conclusions.

2. Motivation

In an aggressive microprocessor design, as more advanced technology become available, existing physical layout is mapped to the new technology in order to boost performance. In most cases, some adjustments in the existing layout are needed in order to maintain design rule correctness of the layout in the new technology. The adjustments are either done by hiring an army of layout technicians to push polygons or by hiring programmers to write special purpose shape processing code to map the design. The first solution is very time and resource consuming. The second solution has its limitation as we will illustrate in the following example. A more general-purpose layout modification technique, that will *minimally perturb* the layout while correcting the design rule errors due to technology changes, is needed.

In an actual microprocessor design project, the spacing distance

ISPD '97 Napa Valley, California USA

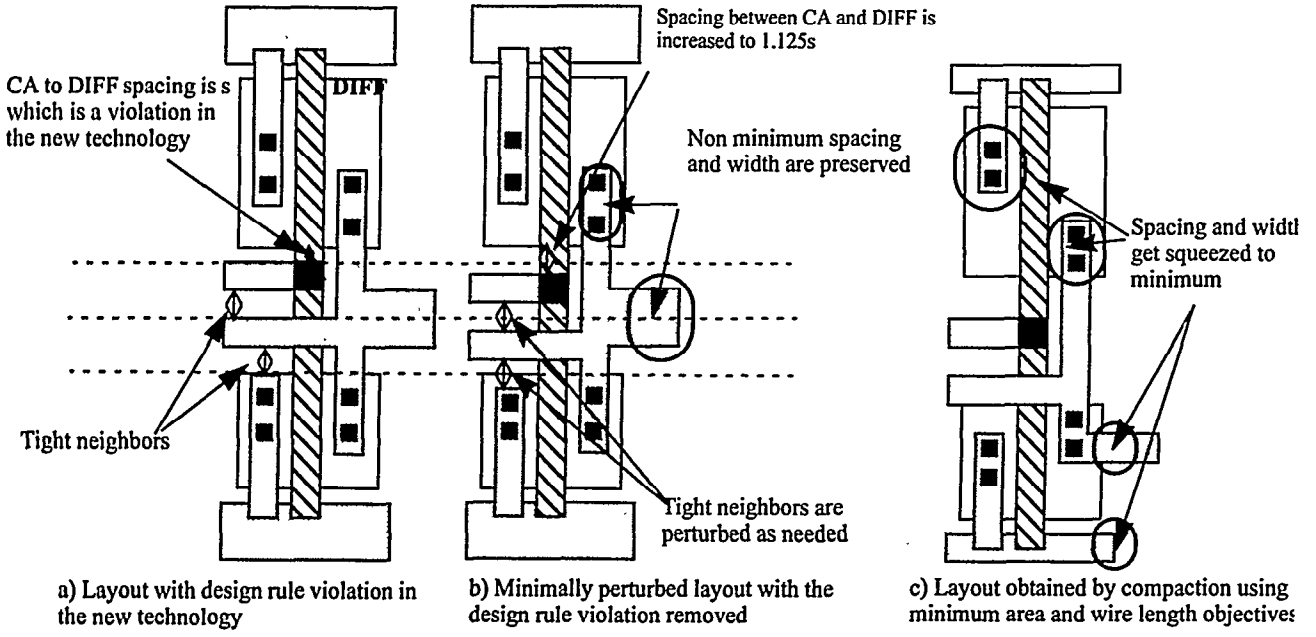

Figure 1 . Contrast of minimum perturbation solution with traditional compaction.

from the diffusion layer, DIFF to the contact layer, CA is changed from *s* to *1.125s* in order to accomplish a linear shrink of the design to a more advanced technology. This single design rule change resulted in millions of DIFF to CA spacing violations. Figure 1 shows a snapshot of a piece of a custom design that illustrate the nature of the problem.

The problem is to change DIFF-to-CA spacing from *s* units apart to *1.125s* units. A manual solution would be to examine if there is room for CA below the violation, and move the CA if possible. When there is no room, tight neighbors of CA must move as well. In some cases this may imply that some device width must be sacrificed, i.e. the DIFF edge moves by *0.125s* units. Special-purpose shape-processing software can be implemented to perform the steps mentioned in the above manual solution. The difficulty lies in capturing the transitive neighboring relationship of tight neighbors and in choosing the changes which least impact the layout.
The constraint graph used by traditional constraint-based compactors [4] can be used to model the layout and capture all the transitive neighboring relationships. However we require a new objective function to produce the desired effect: move the CA shape and its tight neighbors to achieve the new *1.125s* spacing between CA and DIFF, do not move DIFF unless it is necessary and do not move anything that is not affected by the violation. In general, when there are conflicting violations competing for slacks area in the layout, the desired effect is difficult to describe without a formal cost objective.

We define the *perturbation* of a layout object (edges of a shape, wire segments, instances of vias, etc) to be the movement of the object from its existing position. We can describe the above problem informally as: fix all the design rule violations with minimum total perturbation of the layout. In practice, we used weighted perturbation to assign penalty to objects whose movement is less desirable. In the above example, the diffusion edge will have a larger weight than the contact. This helps avoid sacrificing the device width unless it is necessary. Figure 1a shows the DIFF to CA spacing violation due to technology changes. Figure 1b shows the solution using the minimum perturbation criteria, Figure 1c shows the solution obtained by traditional compaction using the minimum area and wire length objective. Notice the disruptive change of the layout in Figure 1c, resulting from the traditional compaction approach. The disruption is very undesirable since the original layout is a working design (in the previous technology).

In the following sections we will describe the minimum perturbation problem and its solution formally.

3. Formulation

In this section we will detail the formulation of the *minimum layout perturbation problem*. For the purposes of this paper, a layout consists of a set of polygons with an assigned layer. We assume that the layout is a Manhattan layout, so each polygon, in turn, consists of an ordered list of horizontal edges $E_X = \{ (Y_i, E_{L_i}, E_{R_i}) \mid (E_{L_i}, E_{R_i}) \in E_Y, i = 1, \ldots, |E_X| \}$ and vertical edges $E_Y = \{ (X_i, E_{B_i}, E_{T_i}) \mid (E_{B_i}, E_{T_i}) \in E_X, i = 1, \ldots, |E_Y| \}$. A horizontal edge can be modified by moving its Y location, and stretched by moving the vertical edges on its ends. Similarly, a vertical edge can be modified by moving its X location, and stretched by moving the horizontal edges on its ends. Ideally, a minimum perturbation should be defined as a 2-dimensional modification of the layout wherein the X and Y coordinates of the edges are modified simultaneously. However it is well known that many 2-dimensional optimization problems of this kind, such as compaction, are NP-hard[10]. We therefore simplify the minimum layout perturbation problem by considering only the *one-dimensional minimum layout perturbation problem*, wherein only one direction of optimization is considered at a time. An approximate solution to the 2D problem is obtained through successive applications of the 1D problem, as is also done in constraint-based compaction.

We next turn to the characterization of the layout constraints. The relative locations of edges are constrained in several ways:

- ground-rules impose minimum separation constraints between edges.
- Connectivity constraints also impose separation constraints.
- Users may require separations on edges, this imposing upper and lower bounds on edge separations.
- To make the problem tractable, we assume that polygon topologies should be invariant. Topological invariance also imposes bounds on the edges.

Note that all of these are two-variable constraints. This observation, coupled with the one-dimensional simplification of the problem leads to a convenient formulation. We will next outline the transformation of the edge location constraints into a constraint graph, and we will consider only the vertical edges. Treatment of the horizontal edges is similar. Let each edge E_i of a layout have a variable X_i. Then a constraint between two edges X_i and X_j is of the form: $X_j - X_i \geq L_{ij}$. Let $G(V,A)$ denote a constraint graph of vertices V and arcs A. Let each edge E_i in the layout have a vertex V_i in the graph, and let X_i represent a variable associated with each vertex. Then for each constraint between two edges E_i and E_j, add an arc A_{ij} between vertices V_i and V_j in the graph. Hence the graph completely represents the set of inequalities that arise from the edge spacing constraints. The construction of a constraint graph is a well studied problem in compaction; for references see [4].

We next turn to the objective of the minimum layout perturbation problem. The minimum layout perturbation problem seeks to minimize changes to a layout, subject to the layout constraints. Hence for a given layout, each vertex X_i has an associated constant, X_i^{old}, that represents the current location of that vertex. We measure the perturbation on layout as a distance function from a given layout to the old (initial) layout:

$$\|X - X^{old}\| = \sum_{v_i \in V} \|X_i - X_i^{old}\| \qquad (1)$$

The set of vertex locations X, which represents a given layout, is said to be feasible if it satisfies all of the constraints in the constraint graph $G(V,A)$. We can now formalize minimum layout perturbation as a constrained optimization problem.

1D minimum layout perturbation problem (MP): Given a layout in the form of a constraint graph $G(V,A)$, where the vertices V have desired locations X^{old}, the problem is to find a set of feasible vertex locations X that minimizes the layout perturbation function:

$$\textbf{\textit{minimize:}} \qquad \|X - X^{old}\|$$

$$\textbf{\textit{subject to:}} \qquad X_j - X_i \geq L_{ij} \quad \forall A_{ij} \in A$$

To complete the formulation, we discuss the choice of perturbation function. Note that if the perturbation function is convex, and since the constraint equations are linear, then the problem is solvable exactly by a variety of methods. There are various possible choices of distance metrics that are suitable for this problem, we have considered the two following metrics:

- L1-metric: weighted absolute value function $\|X_i - X_i^{old}\| = W_i \cdot |X_i - X_i^{old}|$,
- L2-metric: weighted euclidean function $\|X_i - X_i^{old}\| = W_i \cdot (X_i - X_i^{old})^2$.

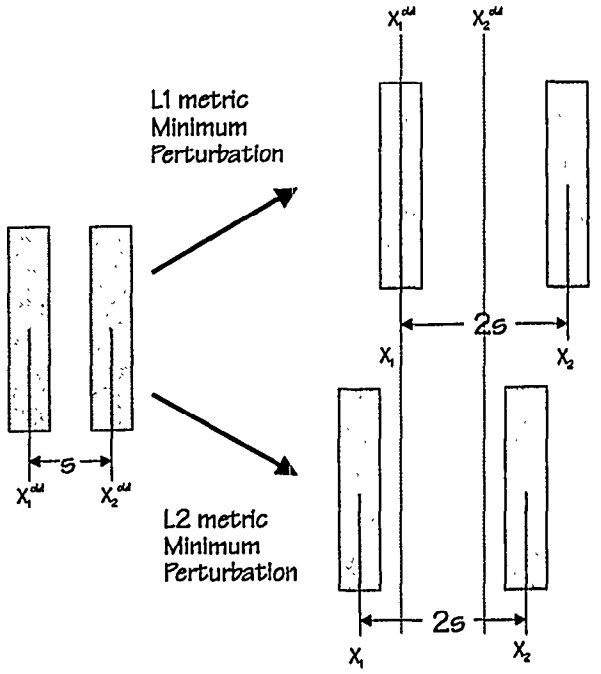

Figure 2 . Minimum layout perturbation using L1 and L2 metrics. Minimum spacing groundrule is 2s, and original layout uses a spacing of s. Note that L2 metric is evenly distributed around the initial layout solution.

An example of a minimum perturbation solution using both metrics is shown in Figure 2. The qualitative difference between these metrics is that the L2 metric tends to distribute the perturbation more evenly than the L1 metric. The practical benefits and disadvantages of each metric are unclear and case dependent. When the L1 metric is used, the problem can be transformed into a linear programming problem that can be solved very efficiently. When the L2 metric is used, the problem can also be transformed similarly but more involved. See Section 4. We use the L1 metric in our implementation and have obtained excellent results. Further investigation is required to determine the benefit of the L2 metric.

4. Algorithms

To characterize the 1D minimum layout perturbation problem, note that the L1 and L2 metrics are convex functions, thus the above objective is convex and must have a global minimum, if feasible. Because of its special structure, the problem can also be solved with the algorithms proposed in [6][7]. In this section, we will focus on the L1 metric problem, and we will show that it can be converted and solved exactly by a transformation to a linear programming problem. Furthermore, we will extend the problem to handle the case where no feasible solution to the original problem exists.

The minimum perturbation objective function can be linearized as follows. Define two variables for each edge E_i, namely L_i, R_i, such that $L_i \leq X_i$, $L_i \leq X_i^{old}$, $R_i \geq X_i$ and $R_i \geq X_i^{old}$. Let V_L denotes the set of new variables L_i's and R_i's. Let A_L denotes the set of arcs representing the new constraints. The resulting linearized constraint graph, denoted $G_L(V \cup V_L, A \cup A_L)$, consists of $3 \cdot |V|$ vertices and $4 \cdot |V| + |A|$ arcs. Then the following linear programming problem is equivalent to the convex 1D minimum layout perturbation problem:

Linear 1D minimum layout perturbation problem (LMP): Given

a layout in the form of set of edges E with locations X^{old}, a constraint graph $G(V,A)$, the problem is to find a set of feasible edge locations X that minimizes the weighted sum:

$$\textbf{minimize:} \quad \sum_{V_i \in V} W_i \cdot (R_i - L_i)$$

$$\textbf{subject to:} \quad X_j - X_i \geq L_{ij} \quad \forall A_{ij} \in A$$

$$L_i \leq X_i,\ L_i \leq X_i^{old} \quad \forall V_i \in V$$

$$R_i \geq X_i,\ R_i \geq X_i^{old} \quad \forall V_i \in V$$

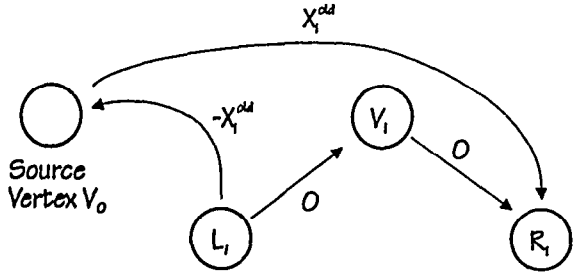

Figure 3. Graph representation of a linearized variable X_i.

The equivalence between these two problems can be shown as follows: consider an optimum solution $\{X, L, R\}$ for *LMP*, where X, L, R denote the sets of values for variables X_i's, L_i's, R_i's respectively. When $X_i \geq X_i^{old}$, $W_i \cdot (R_i - L_i)$ is minimum at $(L_i, R_i) = \left(X_i^{old}, X_i\right)$, thus $W_i \cdot \left|X_i - X_i^{old}\right| = W_i \cdot (R_i - L_i)$. When $X_i \leq X_i^{old}$, $W_i \cdot (R_i - L_i)$ is minimum at $(L_i, R_i) = \left(X_i, X_i^{old}\right)$, again $W_i \cdot \left|X_i - X_i^{old}\right| = W_i \cdot (R_i - L_i)$. Hence, $X \in \{X, L, R\}$ is a solution for *MP* with the same cost. Furthermore, for each optimal solution $\{X\}$ of *MP*, we can construct an equal cost solution for *LMP* with appropriate L and R as above. The linearized graph for a single variable is shown in Figure 3.

The linear 1D minimum layout perturbation problem in practice must be solved while satisfying additional requirements:

I. The layout coordinates X_i must be integers. This is a practical constraint imposed by the structure of industrial layout databases and manufacturing considerations.

II.. Special consideration must be given to layouts where no feasible solution exists to the constraint graph. Producing an infeasible but improved solution is preferable.

Requirement I is handled naturally by the structure of the problem in the case of the L1 metric. It can be shown that if all L_{ij} and X_i^{old} are integers, then the solution of the linear 1D minimum layout perturbation with graph based constraints also consists of integers (*total unimodularity property*)[11]. In the case of the L2 metric, the problem can be made unimodular by converting the quadratic objective function into a piece-wise linear objective, such that the breaks in the objective are all at integer values.
Requirement II implies modifications to the proposed formulation. The problem of handling infeasible problems has been considered previously. In [8] it was shown that the problem of minimizing the number of infeasible constraints is NP-complete. In [8] the problem is approached by constraint relaxation and minimization of a relaxed objective. We take a similar approach. We relax the arc constraints that are not satisfied initially, such that:

- All constraints are feasible.
- A penalty is added to the cost function for the region where the original constraints are not satisfied.
- The current solution, $X = X^{old}$ is made feasible, but not optimal.

We accomplish this by a further transformation of the problem. Let A_e be the set of arcs in A associated with constraints that are not satisfied initially, i.e. $X_j^{old} - X_i^{old} < L_{ij}$. Define a new variable (graph node) M_i for each arc A_{ij} in A_e. Let V_R denotes the set of new variables M_i's. Let A_R denotes the set of new arcs to be added. We construct a relaxed graph, denoted $G_R(V_R \cup V_L \cup V, A - A_e \cup A_R \cup A_L)$, as follows, for each arc $A_{ij} \in A_e$

- Define $D_{ij} = X_j^{old} - X_i^{old}$
- Let $M(i)$ denotes the index of M_i in the vertex set of G_R
- Add constraints: $X_j - M_i \geq D_{ij}$, $M_i - X_i \geq 0$, and $M_i - X_j \geq -L_{ij}$. The constraints are represented by arcs $A_{M(i)j}, A_{iM(i)}, A_{jM(i)} \in A_R$ in the relaxed graph.
- Let A_r denotes the set of arcs $A_{M(i)j}$, i.e. arcs corresponding to constraints $X_j - M_i \geq D_{ij}$

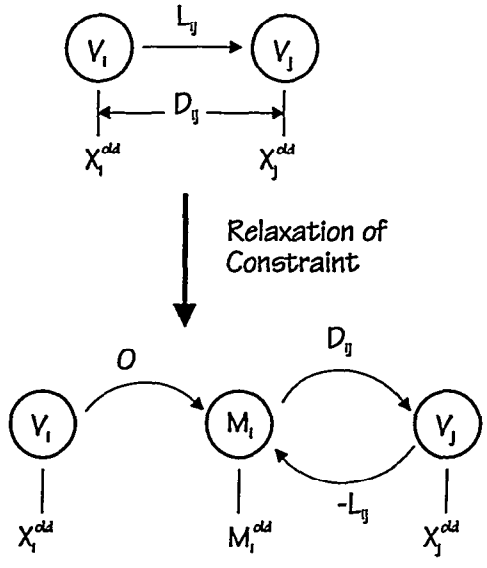

Figure 4. Relaxation of a graph edge yielding an initial feasible solution

We also set the initial location of the relaxed vertices $M_i^{old} = X_i^{old}$. The minimum layout perturbation with relaxed constraints problem can now be formalized as follows:

Relaxed Linear 1D minimum layout perturbation problem (R-LMP): Given a layout in the form of a set of edges E with locations X^{old}, with a set of layout constraints represented by a constraint graph $G(V,A)$, and its corresponding relaxed constraint graph $G_R(V_R \cup V_L \cup V, A - A_e \cup A_R \cup A_L)$, find a set of feasible edg locations X with the objective:

$$\textbf{min:} \quad \sum_{V_i \in V} W_i \cdot (R_i - L_i) + \lambda \cdot \sum_{A_{M(i)} \in A_r} (M_i - X_j + L_{ij})$$

To see that ***R-LMP*** is equivalent to a feasible ***LMP*** problem, first note that $X_j - M_i \leq L_{ij}$, and by setting $\lambda > 0$, $(M_i - X_j + L_{ij}) \cdot \lambda$ is minimum at $X_j - M_i = L_{ij}$, making the contribution of the penalty objective equal to 0. If the value λ is too small, an infeasible solution could have less cost than a feasible solution. We avoid this problem by choosing λ to be greater than $\sum W_i$. This way, for each unit of violation reduction, $\sum W_i \cdot (R_i - L_i) < \lambda$, therefore the contribution from $(M_i - X_j + L_{ij}) \cdot \lambda$ to the cost function will supersede the penalty contributed by movements of object from its existing position. When the contribution from $(M_i - X_j + L_{ij}) \cdot \lambda$ is minimized to zero, unsatisfied constraints in the initial layout are satisfied. Now consider the case when we have an infeasible ***LMP*** problem. The algorithm that solves the ***R-LMP*** problem will return a solution which is not feasible in the original problem. However, since this solution is often an improvement over the original layout, it can be further used to make manual layout changes. Figure 4 illustrates the transformations for a constraint that is not satisfied initially.

The ***R-LMP*** problem can be solved by various algorithms. Fast heuristic algorithms proposed in [7] can be used to solve this problem; unfortunately, the quality of the solution obtained from these heuristics is unpredictable. A general-purpose linear programming solver such as OSL[5], can also be used to solve the problem exactly. However, in this case, the constraint structure of the problem can be exploited to obtain more efficient algorithms. It can be shown that the linear programming dual of this problem is a min-cost network flows problem [9], thus an exact polynomial time solution exists. In practice the best solutions of this problem are obtained by exploiting the graph representation of the problem and applying the Simplex algorithm. The resulting algorithm is called the Graph-Based Simplex algorithm [3] (GBS) or Dual Network Simplex method [9]. For the results presented in this paper we use the implementation of the GBS algorithm detailed in [6].

5. Experimental Results.

We have implemented the Graph-Based Simplex solution in C++. We have tested our implementation on various industrial problems. The examples are all real data from the migration projects that use the minimum perturbation violations removal technique that we implemented. We have used it to accomplish migration from 1.8μ metal pitch technology to 1.2μ technology for the gate-array cells. We have also used it to accomplish a 50% shrink migration for data path macros of a microprocessor. Layout with less than 100 transistors (gate-array book and bit cell of a data path macro) typically takes less than a few seconds to migrate on a PowerPC 604 machine. For a macro with around 5000 transistors, the runtime ranges from 1 to 2 hours.

We also implemented a solution using the OSL linear programming

Cell Name	Problem Size ($\|V\|$, $\|A\|$, Violations)			GBS (# of pivots, secs.)		OSL (secs.)	Ratio OSL/GBS
Layouts with substantial slacks and the perturbations are relatively local							
ERROR1	416	1676	18	222	0.25	4.63	18.5
mux3-H	720	3918	58	1315	1.45	28.81	19.9
BIT_ORG	985	5843	206	1506	1.73	61.42	35.5
bitmid2_org	1184	5853	148	1307	1.38	65.12	47.2
OCD	50727	131125	33	307	549.09	> 4 hrs	N/A
Layouts with less slacks and the perturbations are not very local							
M02-V	170	719	48	433	0.31	1.21	3.9
PG1a-V	538	3344	55	1096	1.33	18.53	13.9
M06-V	607	2974	110	2435	3.46	20.73	6.0
mux3-V	720	3540	60	2436	2.41	24.35	10.1
M06-H	741	5003	169	2698	6.60	45.72	6.9
M07-V	882	6577	62	2399	9.75	61.54	6.3

Table 1: *Empirical results of the minimum perturbation algorithm using the L1 metric objective. The results are compared with the OSL implementation. In these examples the resulting layouts had no violations.*

solver. Table 1 shows the runtime comparison between the two implementations running on a PowerPC 604 machine. Table 1 shows that the GBS solution is 4X to 47X more efficient than the OSL solution. The run-time depends very much on the nature of the data. The GBS implementation tends to be very efficient for layouts that have a lot of slack, where the perturbations are very local. This is because in the Graph-Based Simplex algorithm, the pivoting cost for a local change is very low, while the pivoting cost of a generic linear programing is linear in the number of non-zero elements in the constraint matrix. The first set of examples (ERROR1 to OCD) are layouts with high slacks and low perturbation and the GBS speed up is quite dramatic. For layouts that are generally tight, the efficiency as compared to OSL is less dramatic but still substantial.

For a layout that does not have a feasible solution, our relaxed formulation produces a layout that is as legal as possible. The almost legal layout provides a good starting point so that beneficial local modifications of the layout can be discovered quickly. This is in contrast with techniques that remove unsatisfied constraints from the constraint graph resulting in bad layout topology.

6. Conclusions

We have introduced a novel criterion to legalize a layout with ground rule violations: the minimum layout perturbation criterion. The concept of minimum layout perturbation captures the essence of the ground rule violations removal problem. We have proposed a general formulation for the minimum layout perturbation criterion, which we show can be simplified into a linear programing problem. We then extended the linear programing formulation to handle the case when the layout cannot be legalized, by proposing a constraint relaxation methodology.

We also proposed algorithms that solve the minimum perturbation problem. We noted the special structure of the problem, which we exploited in the implementation of a Graph-Based Simplex algorithm which solves the problem efficiently, as much as 40X faster than an industrial linear program solver.

The algorithms we implemented were used successfully in the migration of data path macros and gate-array library cells. Furthermore these algorithms also been used by custom layout designers to remove violations during the layout process. Other applications of the minimum perturbation technique include incremental device resizing, and wire spacing for noise reduction.

7. Acknowledgments

First and foremost, we would like to thank Rani Narayan for her excellent support of the internal layout compactor for the various migration projects. We also like to thank the following individuals for their various contributions to the development of this work: John Cohn, Dan Ostapko, Wing Luk and Jeff Burns.

8. References

[1] Boyer, D. G. "Symbolic Layout Compaction Review". *Proceedings of 25th Design Automation Conference. June 1988. pp. 383-389.*

[2] Lengauer, T. "Combinatorial Algorithms for Integrated Circuit Layout". John Wiley & Sons. 1990.

[3] Lin, S. L. and Allen J. "Minplex - A Compactor that Minimizes the Bounding Rectangle and Individual Rectangles in a Layout". *IEEE/ACM Design Automation Conference 1986. pp. 123-130.*

[4] Lee, J. F. and Wong, C. K. "A Performance-Aimed Cell Compactor with Automatic Jogs". *IEEE Transactions on Computer Aided Design, Volume CAD-11, Number 12, December 1992. pp. 1495-1507.*

[5] IBM Optimization Subroutine Library, Guide and Reference, Release 2. SC23-0519-03.

[6] Tellez, G. and Sarrafzadeh M. "A Graph-Based Delay Budgeting Algorithm for Large Scale Timing-Driven Placement Problems". *Proceedings of the 5th ACM/SIGDA Physical Design Workshop. April, 1996. pp. 234-240.*

[7] Bamji, C., and Malavasi E. "Enhanced Network Flow Algorithm for Yield Optimization", *IEEE/ACM Design Automation Conference 1996. pp. 746-751.*

[8] Dong, S. K., and Pan, P. and Lo, C. Y. and Liu C. L. "Constraint Relaxation in Graph-Based Compaction". *Proceedings of the 5th ACM/SIGDA Physical Design Workshop. April, 1996. pp. 256-261.*

[9] Ahuja, R.K. and Magnanti, T. L. and Orlin, J. B. "Network Flows Theroy, Algorithms and Applications". *Prentice Hall Inc., Englewood Cliffs, NJ. 1993.*

[10] Kedem, G., and Watanabe H. "Graph-Optimization Techniques for IC Layout and Compaction", *IEEE Transaction on Computer Aided Design, Vol. CAD-3, NO. 1, January 1984. pp. 12-19.*

[11] Christos H. Papadimitriou, Kenneth Steiglitz. "Combinatorial Optimization Algrorithms and Complexity". *Prentice-Hall, Inc. pp 316-318.*